Metamorphosis

Into a New Era

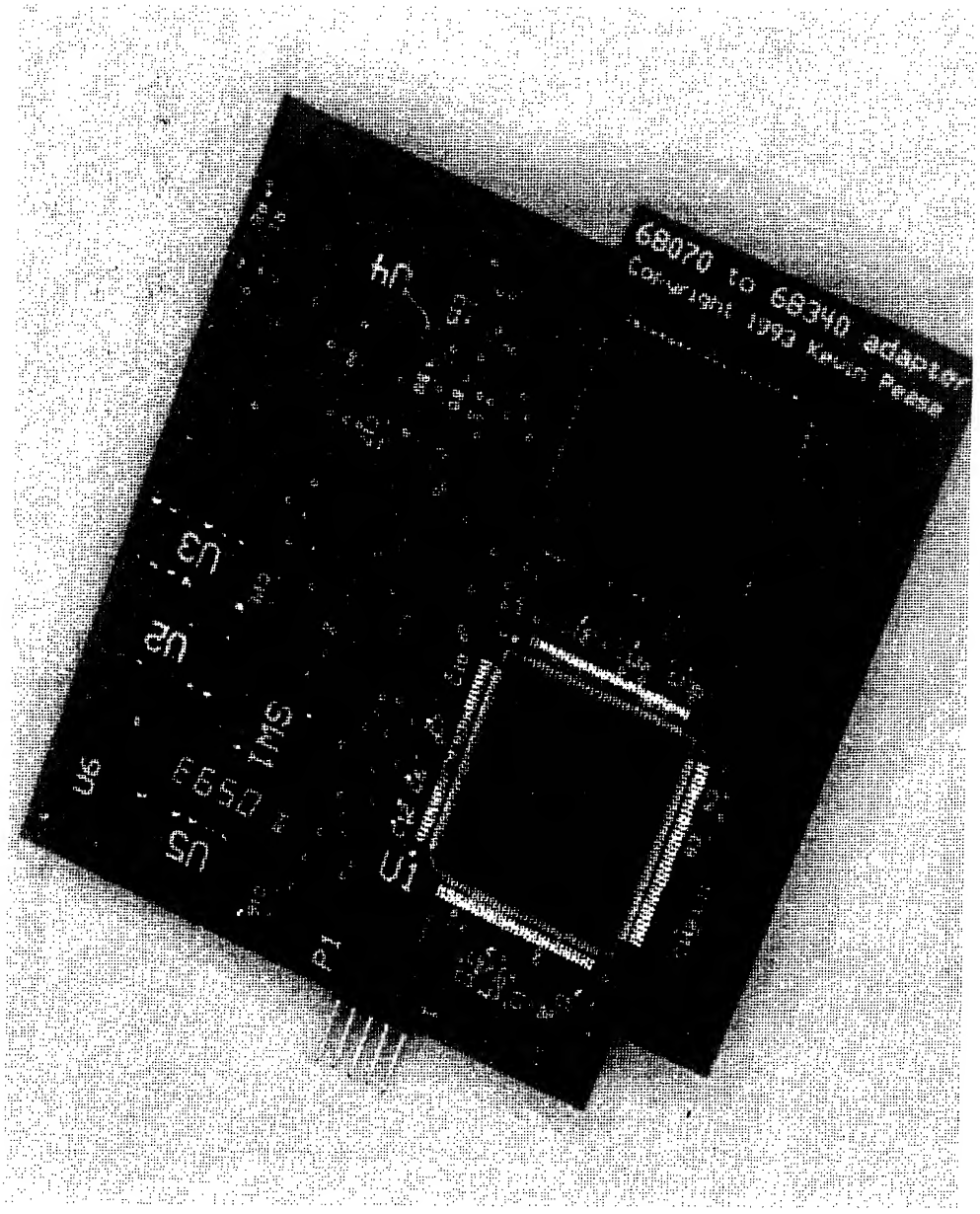
Vol 1, Nbr 5 | For the CoCo/OS9/OSK Communities | October 1993

A Complete OS-9 File System Tutorial

Part IV of The Art of Programming

Variables in BASIC09/MWBASIC

Review: The MM/1 Accelerator



October 1993, Vol 1, Nbr 5

Contents

Metamorphosis
Published monthly by
The Dirt Cheap Computer Stuff Company
1368 Old Highway 50 East
Union, Missouri 63084

Publisher
Mark D. Griffith
Editor
Barbara Ann Griffith
Contributing Writers
Mark Griffith
Tom Kowalski
Shawn Marolf
Don Vaillancourt
David Wordell

Subscriptions
Metamorphosis is published 12 times a year. Subscription rates are $24 per year for the continental United States. Canadian rates are $32 (US) per year. Overseas rates vary. Please write for details.

How To Contact Us
Mail can be sent to the address shown above. Electronic mail can be sent via the Internet to:

76070.41@compuserve.com
or
MARKGRIFFITH@delphi.com

Telephone calls can be made to (314) 583-1168. Please leave a message and your call will be returned.

This publication is composed, formatted, and master pages created entirely on machines running the OS-9 operating system.

Features

Columns

Departments

On the Front Cover

The new MM/1 accelerator board from Kevin Pease. This was scanned on a Macintosh (sorry) at 740 DPI and 256 colors. Converted to a greyscale Postscript image (8.5 megabytes in size!) and printed on a Sun IPC workstation.

From The Editor's Desk

Fall is definitely in the air here in Missouri. The leaves are almost off of the trees, and the ones that haven't fallen yet are beginning to dry out. It's only a matter of time before a good swift wind blows them off. Fireplaces are once again blazing, filling the air with that wonderful smell of burning wood. It's time once again to bring out the winter clothes, put the storm windows up, plan bon fires, and start making chili

The Atlanta CoCo Fest was a huge success for us! Mark enjoyed it immensely, despite the eleven hour drive each way and the bout of flu he came down with the day after he returned. We picked up quite a few magazine subscriptions–thanks. I apologize for not being there myself, but I just started a new job (driving a school bus) and was not able to get the time off. I'm sorry I missed it and will do everything I can to be able to attend the Chicago Fest in May.

We Are All Friends

I would like to make some comments on the subject of ethics in the OS9 community. While our September issue was at the printer, Mark received a complimentary copy of another magazine and saw an article that was submitted to us printed word for word in the other magazine. Obviously, the author submitted the article to more than one magazine at the same time. We apologize. While we are not yet large enough to pay our writers for their contributions, and thus purchase all rights to the article, we still assume our writers will give us the opportunity to print their article before sending the identical article to another magazine to be printed the same month. We hope our readers will not be plagued with this problem in the future.

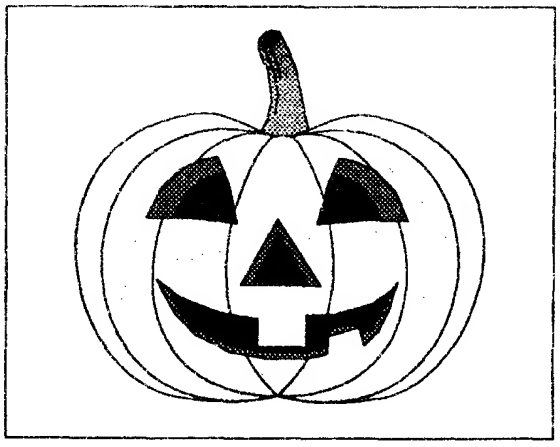
It was also brought to our attention that a large part of one of Mark's articles was reprinted in another magazine, word for word, but without permission or credit given to *Metamorphosis*. While we do not mind if our articles are used by other magazines, we would like to receive credit. Plagarism in any form is a serious offense. Besides, it is just good business ethics not to do this. Hopefully, this was just an oversight on the other publisher's part, and a note of apology will appear in their next issue, with proper credit given.

Our goal, as we previously stated, is to help move OS-9 from it's current status as a "hackers" system into the mainstream of the computing world. We cannot do this if there is any lack of professionalism on the part of the players. We must develop an attitute and working relationship as a knowledgeable, professional group of computer scientists and satisfied users. Anything less will hurt our chances of being accepted into the global community of computer users. Let's please all work together to accomplish this.

Future Plans

Now that the introductory period is over, we can cater soley to our paid subscribers. We have some plans for the future that we hope our readers will find enjoyable. For example, we're in the process of negotiating to get a special area on one of the major telecommunication services to upload all of our articles and source code listings so our readers will be able to download them at their convenience. We're also making arrangements to provide our readers with more indepth articles and product reviews than what we have previously provided. These are just a sampling of the ideas we have in mind to try to make your subscription more meaningful and more enjoyable. We hope you will be pleased with the results.

As always, we hope you will continue to write us. Your suggestions and kudos are most welcome. Have a frightfully funfilled Halloween!



Mail Call!

Subscriptions Keep Coming In

Barbara,

Please sign me up with No Name (which is now named but I cannot remember what). Mark is one of the few who does things instead of talking about them, and I would like to keep track of what he is up to, even if it is OSK. Check enclosed. Your comment to Desertfox leads me to believe there is a stamp collector in your house. A month or so ago we bought some 1960's Science Fiction at the auction at .50 each, and when I read two of them I found some stamps between the pages. Since I have been inactive for 50 years, and the collection was stolen about 10 years ago by one of those buyers, I am including them plus a couple duplicates from my envelope. Good luck with all your projects.

Herbert L. Peterson
Clementon, New Jersey

Say, Herbert. What a nice guy to send us the stamps (and the subscription, of course). We had a great time looking at all of them. My son has stashed them away in his room with the rest of his collection. He was very excited to receive such rare stamps and says to tell you thanks.–Barbara

Dear Mark and Barbara,

After reading the four free issues of your magazine, I am very pleased with what I have seen so far. It's a good feeling to know that there is still strong support out there for the Coco. I am enclosing a check in the amount of $24.00 for a year's subscription to your magazine. Keep up the good work. Thank you.

Lee Stover
New Hampton, Iowa

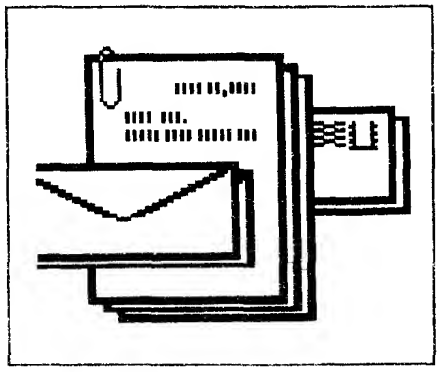
Dear Mark and Barbara,

I've enjoyed the three free issues of the No Name Magazine you provided to illustrate the quality and content you intended to publish. I believe you have a winner! Enclosed find my money order for a subscription. Keep up the good work.

Dave Kelly
Arlington, Texas

Dear Mark and Barbara,

The second issue that I have received of NoName magazine arrived the other day. I definitely want to continue getting it. Thus, a money order for $24.00 is



enclosed. Keep up the good work!

Julian P. Bell
Massachusetts

Mark Griffith,

Enclosed you will find two money orders. The first is for $24.00 for the first year's subscription of your new magazine Metamorphosis.... I do hope everything works out.

Paul R. Myles
Pennsylvania.

Dear Mark & Barbara,

Thank you for the free issues; I found them very enjoyable and informative. I look forward to receiving many more issues in the future with an expanded coverage of the MM1, I hope. I went from a CoCo II to a CoCo III, and now I am currently using an MM1. As you may well imagine, I was a subscriber to the now defunct Rainbow magazine. Please find enclosed a check for $24.00 for a one year's subscription to your new magazine. I wish you great success in your efforts, and hope I will be renewing my subscription to a thriving publication.

James L. Gifford
Kingman, Arizona

To all of you who have decided to stay the course, we thank you very much for your support and subscriptions. We too hope Metamorphosis will be a thriving publication. As we continue to learn as we grow, the magazine will only get better. Keep your letters coming.–Barbara

We goof again!

Mark or Barbara,

Enclosed is my check for a subscription to The NoName Magazine. Also, I would like to point out that my last name is spelled correctly on the mail address label, but in the June 1993 issue it is misspelled. See you in Atlanta, Mark?

William L. Wittman, Jr.
Buffalo, New York

Oops! Terribly sorry about that, William. I am trying very hard to catch those boo-boos, but there always seems to be one or two that get by me. I hope practice makes perfect applies when it comes to proofreading. Thanks for pointing this out.–Barbara

News Clips

Atlanta Success!

The 3rd Annual Atlanta CoCoFest took place during the weekend of the 2nd and 3rd of October. Alan Dages and the Atlanta Computer Society again put on a wonderful show. Attendance was a little low this year than in times past, but those that came brought along their hard earned dollars and they were ready to spend! Many new and exciting products were displayed, both on the showroom floor as well as in the individual rooms after the show closed for the day–products in the making.

One thing that was apparent during the show was that CoCo products were less in demand and the MM/1 and OSK products were the ones that sold. It seems that CoCo Level II support may start dropping in favor of the more advanced systems. Only increased user support can prevent this, or at least slow the progress. Users, support your software and hardware developers! Developers, work on producing more quality products for your customers! Support OS-9! Support the CoCo! Support them all!

Chicago Fest Announced

The 3rd Annual "Last" Chicago Fest was recently announced as a go by the Glenside CoCo Club's president Carl Boll. It looks like the weekend of the 28th and 29th of May, 1994, will be the time, the Holiday Inn in Elgin, IL the place. More news will be posted when available.

In The Next Issue

The November issue will feature several new articles. In addition to Shaun Marolf's excellent series, a review of *Speedisk*, Brian White's new disk optimizer will be featured. The **Hardware Hacking** column will reappear with myself as the feature writer. We'll try many hardware tidbits over the next several months. Hopefully, you'll enjoy all of them. See you then!

Mark Griffith

The Art of Programming

Part IV

Language Syntax

by Shaun Marolf

Shaun Marolf, 30, is a member of the US Navy stationed at Naval Air Station San Diego where he is currently working on his degree in Computer Sciences. He first learned programming in 1979 on an Apple II Plus and is also schooled in electronics and digital circuits. Shaun runs the "Eight Bit Heaven" BBS (619-447-2111) and owns several computers, including an original grey case CoCo 1.

Syntax: 1a. The way in which terms are combined to form phrases and sentences. b. The branch of grammar dealing with the formation of phrases. 2. The rules governing the construction of a machine language.

Syntactics: The branch of semiotics that deals with the formal properties of signs and symbols.

Defining syntax is easy. Explaining it isn't. To begin with, let's discuss the language of the computer. The computer knows only one language, that of binary code. The computer knows the numbers one and zero, (the states of on and off, also called hi and low.) How the computer stores data and programs is a series of these registers. Each register is referred to as a bit. Eight bits put together form a byte.

How a computer translates the information you give it into binary code is done through a translation program. This program is usually in the operating system of the computer. Language translators work by taking the language you are working in and translating it into the binary control codes corresponding to the instruction you give the computer. The problem is the translator requires a perfect set of instructions to be able to give it to the computer in its appropriate binary code. The syntax and syntactics must be absolutely perfect in order for the translator to do its job correctly.

Understanding this, it is important to note that most errors are caused by syntax errors. Now this is not to say you have to make sure that every line is correct before hitting the enter key. That would be asking the impossible. However, it is important to be prepared to go back through the code to find the error.

Each programming language is in itself unique from any other. The C language is considered to be one of the most powerful languages used today, but for all its power, it takes a great deal of time to learn. (Also, for all the hype about portability, it's not quite as portable as it should be.)

Basic is an extremely simple language and takes little time to master. However, it lacks the ability to do some of the more advanced features other languages offer. It also is extremely slow as each line is translated while it's running the program.

There are several other languages as well, each having strengths and weaknesses. Whatever language you do decide to use, you must learn the syntax structure used by it in order for it to understand what you want it to do.

Another problem is that no two versions of any language are the same. Also, there are different standards for languages. Case in point, there is K and R and ANSI standards for C.

This creates several problems for programmers and requires one to decide which version and standard he or she wants to use. Once decided, get all the available information you can on it, from the generic material to the detailed.

Study, learn, follow the examples, etcetera. Remember that methods and techniques constantly change. In order to remain competitive you must keep up with them. Also, if you use a language, such as C, where you need module libraries, locate and acquire as many as possible. Good sources for these are BBS's, networks, programming organizations, and workshops.

Once you start learning how to program, you'll find that it never really stops. Things happen and change so quickly that you'll find yourself constantly learning new things.

Next month we'll discuss the two most used programming structures.

DA-CHART

A program for keeping your children busy

by Tom Kowalski

Tom Kowalski, 46, likes to write BASIC programs on the CoCo and then convert them to BASIC on an Apple and TI-99. While not messing with his computers, he enjoys fishing, gardening, cooking. Tom has served as the doorman at the last two Chicago Fests so many have met him without knowing it. Tom works as a Associate Product Engineer and produces CAD documents on an IBM 5080.

DA-CHART is a short program to print a chore chart on an Epson printer. The minimum requirements needed to run it are a CoCo III, a single disk drive, a serial to parallel converter cable (to go between your CoCo and the printer), and an Epson 24-pin series LQ-500 printer with multiple fonts.

This is a "load and run" type, and, I think, user friendly. I used a single command per programming line approach when I wrote it (it took me over two months). Quite frankly, I got lost trying to figure out what happens in a multiple command line, as many of the CoCo basic programs are written (remember the "one-liners"?). Since this is my first MAJOR basic program, I tried to make it as easy as possible for a novice to read through it and understand what each line does. Also, in a future version (programs are NEVER finished), I want to add programming notes to the lines, to aid others in stepping through it. I also incorporated a "block" or "module" type of programming, as evidenced by the line numbers. This helped me to write, run, and debug the program, and also to understand more fully how the **GOTO** and **GOSUB** routines of basic worked. The program will prompt you for the following information:

```
Child's Name                (line 80)
Month                       (line 120)
Date, begin with Sun        (line 150)
Nbr of Chores (max 8)       (line 200)
Chore Name (max 15)         (line 250)
Money Value of chore        (line 280)
Printer Baud Rate           (line 1130)
Type (Font) Style Cart      (line 2500)
Type (Font) Selection       (line 2650
                             or 3020)
Draft or LQ                 (line 3220)
Print Another Chart
using existing params       (line 5620)
```

You cannot default past any of the prompts in the program by hitting "enter." You must answer the prompts by entering the appropriate letter response, or, if it is a user defined entry, such as "Child's Name", just hit the spacebar a few times, then "enter." If you do not, the program will just loop back to the prompt and wait until you enter correctly.

If you exceed the maximum number of chores, the screen will change color, give you a beep and a reprimand, and then loop back and ask you again. A similar situation will occur if your chore name exceeds fifteen letters.

I tried to offer as many options as possible, based on my limited programming abilities. I already see a few more options and refinements to make to **DA-CHART**, and I will be working on them. A program is NEVER finished!!

I also used color screens and sound to help let the user know what's happening within the program. The program will also check to see if the printer is on-line (lines 1000-10201210-1250). The main print routine body is in lines 5000-5570.

The program will require a full sheet of paper to print on, no matter if there is only one chore or the maximum of eight.

When I demonstrated **DA-CHART** at a meeting of the Glenside Color Computer Club, the members saw many different possibilities and applications for my program. I would like to encourage the users to write programs for the 24-pin printers and, now, more than ever before, for the CoCo, using **DA-CHART** as a platform to start from. I would like to hear from you if you do. If you run into a problem or get stuck, I will certainly try to help, just as I received help and encouragement from fellow CoCoists.

I want to encourage those, as I was encouraged, to try their hand at programming. There's lots of help out there....don't be afraid to ask on your local BBS or at a club meeting. This is what we need to help our orphan CoCo survive and flourish. Enjoy it!

(Source code beings on next page)

```
0 CLEAR 5000: RGB: WIDTH40: CLS6
2 PRINT"*****************************************
4 PRINT*****                                  *****
5 PRINT*****              DACHART             *****
6 PRINT*****     FOR USE WITH THE COCO III    *****
7 PRINT***** & EPSON PRINTER SERIES LQ-500    *****
8 PRINT*****           VERSION 2.0            *****
9 PRINT*****         BY TOM KOWALSKI          *****
10 PRINT*****        104 RIDGE CIRCLE          *****
11 PRINT*****         STREAMWOOD, IL.          *****
12 PRINT*****           60107-1704             *****
13 PRINT*****       COPYRIGHT (C) AUGUST       *****
140 PRINT:PRINT"ENTER DATE, STARTING WITH"
150 INPUT "SUNDAY'S DATE: ";MD$
160 IF MD$= " " THEN MD$= "    "
190 PRINT"ENTER NUMBERS ONLY "
200 INPUT "HOW MANY CHORES ";C
210 IF C>8 THEN SOUND 140,3: SOUND 100,5:GOSUB 500:GOTO 190
220 PRINT
230 CLS6
240 PRINT"CHORE NAME "
250 LINE INPUT CN$(Q)
260 IF LEN(CN$(Q))>15 THEN SOUND 150,2: SOUND 90,2:GOSUB
550:GOTO 240
270 PRINT" MONEY VALUE "
280 INPUT MV$(Q)
290 IF CN$(Q)= " " THEN GOTO 1000
300 IF Q=C THEN GOTO 1000
310 Q=Q+1
320 GOTO 230
330 CLS:WIDTH32:END
500 CLS8
510 PRINT"THIS PROGRAM WILL ACCEPT A MAX."
520 PRINT" OF 8 CHORES TO PRINT CORRECTLY"
530 RETURN
550 CLS1
560 PRINT" *** TOO LONG *** "
570 PRINT" MAXIMUM OF 15 LETTERS"
580 PRINT" PLEASE RE-ENTER"
590 RETURN
1000 CLS6
1010 SP=PEEK(65314) AND 1
1020 IF SP>0 THEN GOSUB 1210
1030 CLS6
1040 '*** PRINTER BAUD RATES **
1050 PRINT"PRINTER BAUD RATES"
1060 PRINT
1070 PRINT" A - 9600"
1080 PRINT" B - 4800"
1090 PRINT" C - 2400"
1100 PRINT" D - 1200"
1110 PRINT" E - 600"
1120 PRINT:PRINT" ENTER LETTER"
1130 K$=INKEY$
1140 IF K$="" THEN 1130
1150 IF K$="A" THEN POKE150,1: GOTO 2300
1160 IF K$="B" THEN POKE150,7: GOTO 2300
1170 IF K$="C" THEN POKE150,18: GOTO 2300
1180 IF K$="D" THEN POKE150,41: GOTO 2300
1190 IF K$="E" THEN POKE150,87: GOTO 2300 ELSE GOTO 1200
1200 SOUND 90,10: GOTO 1130
1210 CLS4
1220 PRINT"PRINTER OFF LINE"
1230 SP=PEEK(65314) AND 1
1240 IF SP <> 0 THEN 1230
1250 RETURN
1500 PRINT
2300 '*** TYPE STYLE FAMILY ***
2310 CLS2
2320 SOUND 110,5
2330 PRINT:PRINT:PRINT"YOU MUST HAVE THE PROPER FONT"
2340 PRINT"CARTRIDGE TO USE OPTIONS."
2350 PRINT"          2 THRU 9"
2360 PRINT
2370 PRINT:PRINT" TYPE STYLE FAMILY"
2380 PRINT:PRINT"    0 - ROMAN"
2390 PRINT"    1 - SANS SERIF"
2400 PRINT"    2 - COURIER"
2410 PRINT"    3 - PRESTIGE"
2420 PRINT"    4 - SCRIPT"
2430 PRINT"    5 - OCR-B"
2440 PRINT"    6 - OCR-A"
2450 PRINT"    7 - ORATOR"
2460 PRINT"    8 - ORATOR-S"
2480 PRINT:PRINT"     DO YOU HAVE THE PROPER CARTRIDGE ?"
2490 PRINT:PRINT"          <Y>ES OR <N>O "
2500 INPUT TSF$
2510 IF TSF$="" THEN GOTO 2500
2520 IF TSF$="N" THEN GOTO 2600
2530 IF TSF$="Y" THEN GOTO 2900 ELSE GOTO 2540
2540 SOUND 130,5: SOUND 110,5:GOTO 2500
2600 CLS7
2610 PRINT:PRINT" TYPE STYLE FAMILY"
2620 PRINT:PRINT"    0 - ROMAN"
2630 PRINT"    1 - SANS SERIF"
2640 PRINT:PRINT:PRINT:PRINT"         ENTER NUMBER"
2650 INPUT TS$
2660 IF TS$="" THEN GOTO 2640
2670 IF TS$="0" THEN X(1)=0: GOTO 3150
2680 IF TS$="1" THEN X(1)=1: GOTO 3150 ELSE GOTO 2690
2690 SOUND 130,5: SOUND 110,5: GOTO 2650
2900 CLS5
2910 PRINT:PRINT"FULL TYPE STYLE FAMILY"
2920 PRINT:PRINT"         0 - ROMAN"
2930 PRINT"         1 - SANS SERIF"
2940 PRINT"         2   COURIER"
2950 PRINT"         3 - PRESTIGE"
2960 PRINT"         4 - SCRIPT"
2970 PRINT"         5 - OCR-B"
2980 PRINT"         6 - OCR-A"
2990 PRINT"         7 - ORATOR"
3000 PRINT"         8 - ORATOR-S"
3010 PRINT"      ENTER OPTION NUMBER"
3020 FTS$=INKEY$
3030 IF FTS$="" THEN GOTO 3020
3040 IF FTS$="0" THEN X(1)=0: GOTO 3150
3050 IF FTS$="1" THEN X(1)=1:GOTO 3150
3060 IF FTS$="2" THEN X(1)=2: GOTO 4050
3070 IF FTS$="3" THEN X(1)=3: GOTO 4050
3080 IF FTS$="4" THEN X(1)=4: GOTO 4050
3090 IF FTS$="5" THEN X(1)=5: GOTO 4050
3100 IF FTS$="6" THEN X(1)=6: GOTO 4050
3110 IF FTS$="7" THEN X(1)=7: GOTO 4050
3120 IF FTS$="8" THEN X(1)=8: GOTO 4050  ELSE GOTO 3130
3130 SOUND 135,10: SOUND 100,8: GOTO 3020
3150 '*** PRINT STYLE ***
3160 CLS5
3170 PRINT:PRINT:PRINT"       PRINT STYLE"
3180 PRINT:PRINT:PRINT
3190 PRINT"    D - DRAFT"
3200 PRINT:PRINT"    L   LETTER QUALITY"
3210 PRINT:PRINT:PRINT"       ENTER LETTER"
3220 PS$=INKEY$
3230 IF PS$="" THEN GOTO 3270
3240 IF PS$="D" THEN GOTO 3270
3250 IF PS$="L" THEN GOTO 3280 ELSE GOTO 3260
3260 SOUND 90,10: GOTO 3220
3270 X(2)=0: GOTO 4000
3280 X(2)=1: GOTO 4000
4000 PRINT#-2,CHR$(27)CHR$(120)CHR$(X(2))
4010 GOTO 5000
4050 PRINT#-2,CHR$(27)CHR$(107)CHR$(X(1))
4060 GOTO 5000
5000 CLS5
5010 PRINT"PRINTING"
5020 PRINT#-2
5040 PRINT#-2,CHR$(27)CHR$(113)CHR$(3);CHR$(27)CHR$(71)
5050 PRINT# 2,TAB(20) CN$ "'S CHORE AND ALLOWANCE REPORT"
5060 PRINT# 2,TAB(25) "WEEK BEGINNING   "M$" /"MD$
5070 PRINT# 2,CHR$(27)CHR$(113)CHR$(0);CHR$(27)CHR$(116)
CHR$(01);CHR$(27)CHR$(54)
5110 X$=STRING$(15,CHR$(205))
5120 Y$=STRING$(5,CHR$(205))
5130 Z$=STRING$(12,CHR$(205))
5140 BX$="                "
5150 BY$="     "
5160 BZ$="          "
5170 PRINT#-2,TAB(2)CHR$(201)X$CHR$(203)Y$CHR$(203)
Y$CHR$(203)Y$CHR$(203)Y$CHR$(203)Y$CHR$(203)Y$CHR$(203)
Y$CHR$(203)Y$CHR$(203)Z$CHR$(187)
5180 M1$="     CHORE      "
5190 M2$=" SUN "
5200 M3$=" MON "
5210 M4$=" TUE "
5220 M5$=" WED "
5230 M6$=" THU "
5240 M7$=" FRI "
5250 M8$=" SAT "
5260 M9$=" TOTAL AMT. "
5270 MX$="$ AMT"
5280 PRINT#-2,TAB(2)CHR$(186)M1$CHR$(186)MX$CHR$(186)
M2$CHR$(186)M3$CHR$(186)M4$CHR$(186)M5$CHR$(186)
M6$CHR$(186)M7$CHR$(186)M8$CHR$(186)M9$CHR$(186)
5290 PRINT#-2,TAB(2)CHR$(204)X$CHR$(206)Y$CHR$(206)
Y$CHR$(206)Y$CHR$(206)Y$CHR$(206)Y$CHR$(206)Y$CHR$(206)
Y$CHR$(206)Y$CHR$(206)Z$CHR$(185)
5300 R=1
5310 FOR Q=1 TO C
```

```
5330 PRINT#-2,TAB(2)CHR$(186)CN$(Q)TAB(18)CHR$(186)
TAB(19)MV$(Q)TAB(24)CHR$(186)BY$CHR$(186)BY$CHR$(186)
BY$CHR$(186)BY$CHR$(186)BY$CHR$(186)BY$CHR$(186)
BY$CHR$(186)BZ$CHR$(186)
5340 PRINT#-2,TAB(2)CHR$(186)BX$CHR$(204)Y$CHR$(206)
Y$CHR$(206)Y$CHR$(206)Y$CHR$(206)Y$CHR$(206)Y$CHR$(206)
Y$CHR$(206)Y$CHR$(206)Z$CHR$(185)
5350 PRINT#-2,TAB(2)CHR$(186)TAB(18)CHR$(186)TAB(24)
CHR$(186)TAB(30)CHR$(186)TAB(36)CHR$(186)TAB(42)CHR$(186)
TAB(48)CHR$(186)TAB(54)CHR$(186)TAB(60)CHR$(186)TAB(66)
CHR$(186)TAB(79)CHR$(186)
5360 PRINT#-2,TAB(2)CHR$(186)BX$CHR$(204)Y$CHR$(206)
Y$CHR$(206)Y$CHR$(206)Y$CHR$(206)Y$CHR$(206)Y$CHR$(206)
Y$CHR$(206)Y$CHR$(206)Z$CHR$(185)
5370 PRINT#-2,TAB(2/CHR$(186)TAB(18)CHR$(186)TAB(24)
CHR$(186)TAB(30)CHR$(186)TAB(36)CHR$(186)TAB(42)CHR$(186)
TAB(48)CHR$(186)TAB(54)CHR$(186)TAB(60)CHR$(186)TAB(66)
CHR$(186)TAB(79)CHR$(186)
5380 IF R<0 THEN GOTO 5500
5385 R=R+1
5390 PRINT#-2,TAB(2)CHR$(204)X$CHR$(206)Y$CHR$(206)
Y$CHR$(206)Y$CHR$(206)Y$CHR$(206)Y$CHR$(206)Y$CHR$(206)
Y$CHR$(206)Y$CHR$(206)Z$CHR$(185)
5400 NEXT Q
5500 PRINT#-2,TAB(2)CHR$(200)X$CHR$(202)Y$CHR$(202)
Y$CHR$(202)Y$CHR$(206)Y$CHR$(202)Y$CHR$(202)Y$CHR$(202)
Y$CHR$(202)Y$CHR$(206)Z$CHR$(185)
5510 M%$=" TOTAL AMOUNT FOR THE WEEK"
5520 B$="    BONUS"
5530 PRINT#-2,TAB(36)CHR$(186)M%$TAB(66)CHR$(186)TAB(79)
CHR$(186)
5540 PRINT#-2,TAB(36)CHR$(200)Y$CHR$(205)Y$CHR$(205)
Y$CHR$(205)Y$CHR$(205)Y$CHR$(206)Z$CHR$(185)
5550 PRINT# 2,TAB(54)CHR$(186)B$TAB(66)CHR$(186)
M%CHR$(186)
5560 PRINT# 2,TAB(54)CHR$(200)Y$CHR$(205)Y$CHR$(202)
Z$CHR$(188)
5570 PRINT#-2,CHR$(12)
5600 PRINT:PRINT:PRINT"      PRINT ANOTHER"
5610 PRINT:PRINT"           CHORE CHART ?"
5615 PRINT:PRINT"                YES OR NO"
5620 LINE INPUT PA$
5630 IF PA$="" THEN END
5640 IF PA$="N" THEN END
5650 IF PA$="Y"; THEN GOTO 5000
```


From The Jargon File

Brute Force *adj.* - Describes a primitive programming style, one in which the programmer relies on the computer's processing power instead of using his or her own intelligence to simplify the problem, often ignoring problems of scale and applying naive methods suited to small problems directly to large ones.

The (canonical) example of a brute-force algorithm is associated with the 'traveling salesman problem' (TSP), a classical problem: Suppose a person is in Boston, and wishes to drive to N other cities. In what order should he or she visit them in order to minimize the distance travelled? The brute-force method is to simply generate all possible routes and compare the distances. While guaranteed to work and simple to implement, this algorithm is clearly very stupid in that it considers even obviously absurd routes (like going from Boston to Houston via San Francisco and New York, in that order). For very small N it works well, but it rapidly becomes absurdly inefficient when N increases (for N = 15, there are already 1,307,674,368,000 possible routes to consider, and for N = 1000 - you get the idea.

A more simple-minded example of brute-force programming is finding the smallest number in a large list by first using an existing program to sort the list in ascending order, and then picking the first number off the front.

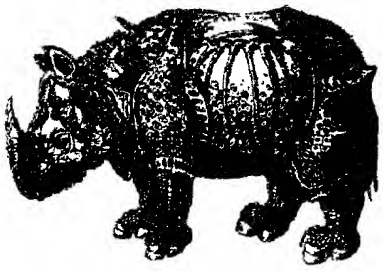
Whether brute-force programming should be considered stupid or not depends on the context; if the problem isn't too big, the extra CPU time spent on a brute-force solution may cost less than the programmer time it would take to develop a more 'intelligent' algorithm. Alternatively, a more intelligent algorithm may imply more long-term complexity cost and bug-chasing than are justified by the speed improvement.

Ken Thompson, co-inventor of UNIX, is reported to have uttered the epigram "When in doubt, use brute force". He probably intended this as a (ha ha only serious), but the original UNIX kernel's preference for simple, robust, and portable algorithms over (brittle) 'smart' ones does seem to have been a significant factor in the success of that OS. Like so many other tradeoffs in software design, the choice between brute force and complex, finely-tuned cleverness is often a difficult one that requires both engineering savvy and delicate esthetic judgment.

Fighting The Hard Drive Monster

Adventures with a PT-68K

by David Wordell



I recently bought a case and power supply from the local computer flea market. Got a good deal on it, too. A real nice case and 200 watt power supply for just $40.00. I then ordered the PT 68K-4 Single Board Computer from Peripheral Technology, Marietta, Georgia. I also ordered Professional OS-9/68000, version 2.4, by Microware, from the same company. I also bought a 3 1/2 inch 1.44 Meg Floppy Drive and installed this in the new case. Got it at the same flea market for $43.00. When the circuit board arrived, I put it in my new case and plugged in the power supply cables. I also installed the little cable which brings Com-1 to the rear of the case so that I could hook the CoCo to it and see what this baby could do. Everything went well until I hooked the Floppy Drive cable backward on the mother board. Scratch one boot disk. Actually, not that bad. It just messed up two files. It so happened that they were hdsetup and hdformat. Now, after getting the cable hooked up correctly, I could boot up but was unable to get the hard drive running. Oh yes, I had a 72 Meg, full height hard drive on one of my CoCo's that I decided to install on this machine. This was a minor inconvenience, and Frederic Brown of Peripheral Technology graciously sent me another bootdisk.

The new bootdisk soon arrived, and I set out to install the hard drive. Following the instructions, I created a new /h0 descriptor for it and proceeded to format it. The key phrase here is "following the instructions." All went well. The drive was ready, and I used Dsave to copy the three disks full of OS-9 goodies to it. By the way, I found many familiar names of utilities that we have on the CoCo and, to my surprise, many have improved features. I felt really "at home" with this new OS-9/68000.

Let's go over what I had up to this point. The mother board is installed . There is a cable with a female DB-25 connector coming to the rear of the case to hook the CoCo to. There is a 3 1/2 inch, 1.44 Meg floppy drive installed. There is a 72 Meg hard drive installed. To run this machine, I power up the CoCo and run SuperComm. I set the baud rate to 9600 and the terminal type to OS-9. Then I power up the PT 68K-4 and a menu appears. I choose number 4, Boot from Floppy, and I'm off and running.

I happen to be taking a course in C programming so I had to try out the C compiler. The first problem I encountered was that unless I decided to retype all my source code, I didn't have any way to get these files to the new machine. I finally did find a way, but it is not pretty. First, chd to the ram disk on the new machine. Then type "build source.c" and the familiar ? prompt will greet you. Then flip to another window on the CoCo and list the file source.c and redirect it to SuperComm's serial port, by typing "list source.c >/t2". Return to the window running SuperComm and you see build taking the text as if you had typed it in directly. There was a problem, however. Each line ended in 0D0A. OS-9 prefers 0D only. I found this out by using the Dump utility. Works just like the one on the CoCo. This made for a messy listing, not to mention the C compiler getting all choked up. I looked in an ASCII conversion chart that was in my printer manual for the value of OD and OA. I found that OA was called Ctrl-J in that chart. Guess what? By using edt, just like edit on the CoCo, I could get rid of the "Ctrl-J's" by typing "c*/J//<enter>" and they all disappeared in a flash. The file now looked like it should. This was workable, but I felt I could do better.

My CoCo presently has a 3 1/2 inch, 720K drive as /d0. I had an extra 3 1/2 inch, 720K drive that I decided to install in the PT 68K-4 case. I intended to make a CoCo compatible descriptor and

then be able to swap disks with my CoCo, which would make things much easier than the above procedure had been. I installed the drive and proceeded to make a new descriptor for it. I was using the same procedure that I had used to make the hard drive descriptor earlier. I had found the d1.a file in the IO directory and edited it to be for a 720K drive following the instructions in the file. Then I used the Make utility to create a new descriptor. I kept getting "Target file not found" error. This couldn't be. This had worked before on h0.a. Why not now? I decided to try making a new hard drive descriptor, just to test my syntax you understand. Same error, "Target file not found." Okay, logic has to kick in here somewhere. When I had first made the hard drive descriptor, I had been working with only a floppy disk system. I went back to the disks, you know, chd /d0/IO and chx /d0/CMDS, sort of make believe that the hard drive is not there. Guess what? IT WORKED! Unbelievable.

Between the time that I bought and assembled the PT 68K-4 and the time I decided that I needed the CoCo compatible drive, I had attended the Middle America Fest 93, which was presented by the Mid Iowa & Country CoCo Club, in Des Moines, Iowa. I had received a disk from someone, whom I hope will forgive me for not remembering his name, which contained the OS-9/68000 version of Dmode. It seemed pretty obvious to me that I had failed to put something on my hard drive that certainly was on my disks. Probably due to the fact that I had one original version of the #1 disk that had messed up files on it and a second version of the #1 disk, which replaced those files, but may have not had everything else on it. I decided that I would format the hard drive and reload everything. It was only three disks. I had purchased Ved and Vprint but not yet installed them. I also purchased the demo version of G-Windows but had not installed that one either. It was just the three system disks. There should be no problem, and "I better get everything in there working now before adding more and more files" I reasoned.

I booted up with the original disk, no hard drive needed. I copied Dmode to the commands directory of that disk so I could use it. Okay, here we go, "dmode /h0<enter>" and up comes the dmode report for /h0, just like on the CoCo. I was in seventh heaven. My drive has 754 cylinders and 11 heads. It had served me for several years on one of my CoCo setups. The dmode report showed that the stock descriptor had 614 cylinders and 4 heads. Everything else was the same. I typed "dmode /h0 cyl=2F2 sid=0b<enter>". That should do it. I then typed "format /h0<enter>" and the format

"Then I came to the "Do you want a logical format" question. I said "Yes" and wham!"

utility came up and showed me all the data it was going to use to format this drive. It also warned me that it was a hard drive. I plodded forward. It asked "Did I want a physical format" and I answered "Yes." Off it went. Some time later it asked for a volume name and whether I wanted a "Logical format". Again I said "Yes" but then in an instant an error flashed on the screen. "Fatal error reading logical sector zero"! Now what was this? Let's try this again. "Format /h0<enter>" etc., etc. Bingo, the same error. Had I messed up my hard drive when I installed the new floppy drive? Had I jiggled the head on sector zero, making it unreadable? Could I be that unlucky? After several hours of checking and double checking, I concluded just that. It was time to buy another hard drive.

I went to a local discount, used, computer junk, type store looking for a bargain. I found another 72 Meg hard drive. This one happened to be a half height drive. That would even fit into the case I had a lot easier. The whole thing would end up lighter also, a consideration when you bring your system to the club meeting every month. This drive also presented one other problem. It had 1170 cylinders and 7 heads. These are MFM drives using XT hard drive controller cards just like the Burke & Burke CoCo XT-RTC adapter I had on my CoCo. In fact, the hard drive controller card I was using had come from the same CoCo setup that the first hard drive had come from. I knew that this meant that I could only use 1024 cylinders, a limitation of the XT cards, not a limitation of OS-9. I wasn't happy, but I could live with that.

I removed the original full height hard drive and replaced it with the new half height. I got everything plugged in, double checked everything, and fired it up. I booted, as before, with the disk system, using the CoCo as the terminal. So far so good. I typed "dmode /h0<enter>" and the same report I had seen before came up. This time I typed "dmode /h0 cyl=400 sid=07<enter>". Looking good, so I typed "format /h0<enter>" as before. Same questions, same answers, everything seemed to be working correctly. Then I came to the "Do you want a logical format" question. I said "Yes" and wham! "Fatal error reading logical sector zero"! Now I ask you, do you believe that I could have bought another drive that just happened to have a bad logical sector zero? Naw, well . . ., naw, couldn't be . . . could it?

I was getting desperate. I placed a call to Peripheral Technology. Could Frederic be there on a Saturday? Maybe my luck was about to change. Frederic was there. I whined, I cried, I tried to explain. Frederic concluded from my explanation, full of assurances

that "I had been using OS-9 for years, I knew what I was doing. I couldn't be doing it wrong" that maybe my hard drive controller card was bad. It sounded reasonable to me. With many thanks, I hung up and got ready to go out and buy another hard drive controller card. Not an easy task on a Saturday afternoon, but I found one.

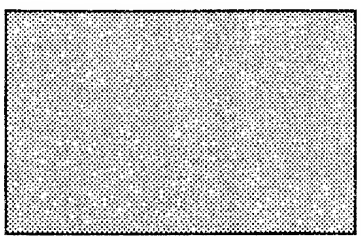
Here we go again. I was lucky once more. I got a hard drive controller card that was the same model as the original card that I was using. Easy to know how to configure the jumpers that way. I did have the documentation for that model. Unfortunately, same story, same day, same results. "Fatal error reading logical sector zero" came back to haunt me. I was really pulling my hair out. "This is not possible," I said over and over. This was like a roller coaster ride that I didn't want to be on. I needed a plan and I needed it fast.

The original 72 Meg, full height, hard drive had been in a CoCo system, working, just a few weeks ago. The whole thing was still sitting here on the table. I decided to put it all back together and see what happened. I plugged the original hard drive controller card into the Burke & Burke CoCo XT-RTC and connected the cables for the hard drive. I then reassembled the case. I put the hard drive back into its case and plugged in the power supply cable and the controller cables from the Burke & Burke. I had a boot disk, just for this occasion, that would boot without need for the hard drive but does have the hard drive descriptors in memory. In fact, I have one of these for each of my systems. I fired it up. All went well. The system came up running. I was a tad impatient by this time and knew it would take a long time to format this drive on the CoCo system. I didn't want to wait. I used dmode to change to 10 cylinders, 4 sides. I typed "format /h0<enter>". In a very short time I got the familiar question and gave my answer. "Yes, I did want a logical format", at least I hoped I did. Bing, bing, bing, just like clockwork, the drive stepped out and formatted and my prompt returned. I apprehensively typed "dir /h0<enter>". I got the familiar report of an empty directory on a freshly formatted disk. Fantastic, it had worked. The drive was okay. Could the new drive also be okay? Just a few minutes swapping the two drives, and I had the answer, YES! I had not wasted my money. The new drive was good. Could the new hard drive controller card be good also? You bet, good as gold.

Three days previous, I had placed a call to my very good friend Lee Veal. We have traveled to several Fests together and spent many hours in the car discussing OS-9. He has an excellent understanding of OS-9 and many years of experience to go with it. He had been out of town, and his wife had given him the message when he returned. He called me Sunday evening. I told him this sad story that I have just told you. By the time we got to this part, he was laughing out loud. In fact, he was roaring. I didn't see anything funny at all. I was beginning to think that there was something wrong with my new computer, and that was not something I wanted to hear. When Lee finally calmed down, he had the answer. Remember the key phrase following the instructions earlier? Well, I had swerved from this road just a little. I had decided to use Dmode to set the descriptors. After all, once I set them, I could have saved them to the BOOTO-BJS directory for making a boot-disk using Os9gen. For those of you who do not have OS-9/68000, this is like the MODULES directory on the CoCo. I knew this would work. What I forgot is that Dmode expected me to be smart enough to realize that computers start counting from zero (0) and humans start counting from one (1). My mistake, pointed out by my friend Lee, had been two foal but caused the same error. On the first drive, 754 had been the actual last cylinder. The real end of the line. The computer would call that 753. Since I called it 754, when the head got to the end it did the next logical thing. It went right back to sector zero. Somewhere on the drive circuitry there must be a counter that says 753 plus one, rolls right back around to zero. When I formatted the drive, the last track written was actually on top of the first track written, logical sector zero, effectively wiping it out by writing data for track 754 on top of it. When format tried to check out the format and write the logical data to the tracks, it found no sector zero to write to.

The second drive had suffered from a similar problem. Even though it had less sectors than I specified and could not have rolled its counter, the hard drive controller card also has a counter. It has a limit of 1024 or actually zero (0) to 1023. The same thing happened. When the drive got to track 1024, the hard drive controller card sent it back to track zero, effectively creating the same error as on the first drive. It also wrote right over the label for track zero. It looked like both drives were bad.

Needless to say, this story has a happy ending. The PT 68K-4 is up and running, with the newer hard drive. The original, full height will be the backup or spare. I am now progressing well with the PT 68K-4. In fact, although I started this story on the CoCo, using Ved, I have completed it on the PT 68K-4, also using Ved, the 68000 version. If there is any interest, I will write more stories about OS-9, the CoCo, and the PT 68K-4.



The Soul of a New Machine

The 68340 Accelerator Board for the MM/1

by Mark Griffith

The MM/1 personal computer from Interactive Media Systems has been available for exactly three years since making its debut at the Atlanta CoCoFest in October of 1990. A little over a year ago, rumors started that a "speedup" board was in the works to replace the Motorola 68000 compatible 68070 microprocessor. During the Chicago fest in May, Kevin Pease, the designer of the MM/1, showed up with one of his accelerator boards. In the last five months, Kevin and Carl Kreider have been working to bring the accelerator to the public. Now, they are available and were sold at the Atlanta CoCoFest. I asked Carl if I could get an advanced look at one of the boards, and he had Kevin Pease send one to review.

A few days later, a small box arrived with the accelerator board, a couple of ROMs, one PAL, a disk, and some instructions. The board itself, as seen on this month's front cover, is surprisingly sparse. There are only a few chips and other components, the main piece being the new Motorola 68340 CPU. This CPU is for the most part the equivalent of a 68020. There are some neumonics that are missing from its instruction set, but they are ones that are hardly, if ever, used. These are shown in table 1 below.

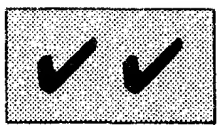
Instruction	Description
bfchgclr	
bfexts	
bfextu	
bfff0	bitfields
bfins	
bfset	
bftst	
callm	call module
rtm	rtn from module
cas	compare and set
cas2	compare and set
pack	pack BCD string
unpk	unpack string

In addition, no coprocessor instructions are included either.

The accelerator kit comes with many new modules to take advantage of the increased speed. A list is shown on the next page. The new SCSI modules are designed to take advantage of the increased DMA capabilities of the 68340. The hard drive descriptors are for use with the new SCSI drivers. The new floppy disk driver (rb37c65) and the new serial I/O modules (sc68681, sc68901, and sc68340) include several bug fixes from the last IMS updated versions. Also included is a new windio module, version 50, and new window descriptors. I'll have more to say about this new version later.

As noted in the included instructions, you must make a new boot disk with the new modules BEFORE installing the accelerator board. If you do not, you won't be able to boot from your old boot disk. The instructions I received were not the final copy and did not provide enough information to make a bootable disk. This resulted in my having to de-install the accelerator board, reinstall the old ROMs, put the boards back together, reboot under the 68070, and then make another accelerator boot disk. I had to do this several times to get it right. One thing that was not mentioned was that the new **stdfont** module must be included in the bootfile, as well as the **cio** and **math** modules. This has been corrected in the final version of the included documentation by giving an example bootlist file of which modules to include.

Installation of the accelerator board was simple. I removed the I/O board and used a special PLCC device puller to remove the 68070 from the socket. The included instructions describe how to make a chip puller from a large paper clip, but I preferred to use the puller. Although the paper clip puller will work, I would recommend using extreme caution when doing this if you ever want to reuse the 68070 again. Kevin Pease yanks them out like a dentist pulling teeth, but he is apparently luckier than I am.



I then inserted the accelerator board into the empty socket and removed and replaced the two ROMs with the new versions. This brings up another caution. IMS originally shipped their distribution disks with some OSK version 2.3 commands installed, notably the format utility and the cio trap handler. Users never noticed this because the version 2.4 modules were also included in the ROMs along with a host of other well used utilities, like COPY, DIR, and so on. Of course, OS-9 would always use the modules already in memory rather than load an older one from disk. The problem is, the accelerator ROMs do not include anything other than the boot code. No utilities or trap handler modules, and no debugger.

If a user installs the accelerator board and ROMs, boots up, and tries to format a hard drive with the version 2.3 format utility, it won't work. The same applies to any applications that used the cio trap handler or old math modules. To make sure this doesn't happen, you need to check all the modules in ROM on your 68070 machine with the same modules in your commands directory. If they are not the same, save the module in memory to disk using the SAVE utility. Once you have made sure all your commands directory modules are the latest version, you can proceed

After installing the ROMs, I reassembled the two boards and found that the I/O board could not be securely fastened to the mother board standoffs without flexing it too much. The accelerator board protruded too far from the socket. After speaking to Kevin Pease about this, we found that by breaking off the four little seats in the bottom of the 68070 socket, the accelerator board would fit deeper into the socket and allow the two boards to be connected again without undue flexing. However, if you do this, don't plug your 68070 back into that socket or it will fit so deeply inside you might not get it back out again! I found out and if it were not for that special PLCC chip puller, I'd be stuck running an '070 system.

New Modules				
cio	dd.h0	dd.h0a	dd.h0fmt	
h0	h0a	h0b	h0c	h0d
h0e	h0f	h0fmt	h0g	h1
h1a	h1b	h1c	h1d	h1e
h1f	h1fmt	h1g	h2	h2fmt
h5fmt	h6	h6fmt	h7	h7fmt
init	kernel	keydrv	math	msdrv.901
p	rb37c65	rbvccs	sc68340	sc68681
sc68901	scsi_mm1a	stdfonts	t0	
t5	term	w	w0	w1
w10	w11	w12	w13	w14
w15	w2	w3	w4	w5
w6	w7	w8	w9	windio

Once I had everything installed correctly, I powered up the machine and noticed the new boot prompt also includes the speed at which the CPU is running. In this case, I started with the lowest setting, 11.98 Mhz. As I mentioned before, once I got all the correct modules included in the boot file, the machine booted normally.

The first thing I noticed was my mouse no longer worked. The keyboard mouse didn't work either. Apparently, Kevin Pease has a Logitech mouse and the mouse driver only supports that model. I have a Logitech compatible mouse and it does not work with the new driver. Kevin was not aware of this problem and is in the process of fixing the driver. To provide a temporary fix, I unplugged the mouse from the /t2 socket and the keyboard mouse then worked. Also, the current mouse driver is set to work only on the /t2 port.

There were some other problems. The new version of windio was "broken" again and many graphics applications did not work. The problem seemed to be with changing palettes. Kevin Pease was aware of this and has already contracted someone to work on it. In addition, there is no sound driver for this new version, so even the keyboard beep doesn't work. Again, someone has been contacted and new drivers should be available soon. As of this writing, I do not know the date when they will be available.

Now that the system was running, I could perform some simple tests. I decided to run tests that were easily measured, but still reflected tasks that would be done during everyday operation of the machine. All of these tests involved memory intensive operations, since disk or serial I/O would not be effected greatly, if at all. The chart below shows the results of the tests at four different CPU speed settings. I included a dhrystone test just as a means of plotting raw performance.

The **extract** test involved using the *lha* utility to extract a rather large archive of files. The

Test	68070	11.98		14.75		16.59		20.28	
Dhyrstone	1020	1612	1.58	1666	1.63	2083	2.04	2777	2.72
Extract	71	50	1.42	43	1.65	41	1.73	37	1.92
Archive	192	125	1.53	105	2.83	86	2.23	76	2.53
Compile	400	253	1.58	213	1.88	188	2.13	163	2.45
List	136	130	1.05	104	1.31	96	1.42	93	1.46

archive test was archiving all the files just extracted. The **compile** test was using the standard 'C' compiler to compile all the sources for Sterm 1.5.1. The **list** test was simply listing a 58k file to the screen. All the test results, except for the dhrystone test, are displayed in elapsed seconds. The dhrystone test results are the number of dhrystones completed in one second.

At 11.98 Mhz, the MM/1 was slightly faster, but nothing to brag about. Since the stock MM/1 runs at 15 Mhz, I decided to try a CPU speed as close to that as possible, 14.75 Mhz. I then tried 16.59 Mhz since this is the recommended setting for the production units. The 20.28 Mhz tests were done to test the ability of this increased speed on the components of the I/O board. I was able to run at this speed because the unit I was testing was the original prototype and had a 68340 installed that was rated at 25 Mhz. Production units are all rated at 16 Mhz. Increasing the CPU speed above that, while it is possible, is not recommended by the designer.

The "feel" of the MM/1 with the accelerator installed and running at 16 Mhz made it like a new machine. Applications snapped on the screen and flipping through pages on the text editor was lightning fast. Everything felt crisper, smoother, and more powerful. What graphics applications I was able to test were substantially quicker.

In spite of the initial problems and the work that is still needed on the software, the accelerator board is a much needed addon for any MM/1 owner. Kevin Pease and Carl Kreider are committed to providing the customer support today's users are accustomed to having. Most of the problems identified were due to the rush in getting the product to the customers. As is too often the case in this market, the developers are doing this in their spare time *(if there is such as thing-Ed.)* and not as much time is devoted to testing as should be. You can't blame them for trying to keep up with the larger developers, considering the work load they carry and what they are trying to do.

Why should someone buy an accelerator? Well, if you are a developer, the increased speed of compiles will save you much time. For those working on graphics or sound processing, a 2.5 time increase in speed would work wonders. For the average user, although there would be no increased speed in serial I/O or disk access, the more advanced SCSI and serial drivers would mean increased reliability.

Both Kevin Pease and Carl Kreider should be praised for their efforts. A lot of time, energy, and money when into development of this device. Putting the problems mentioned aside, this is a quality product with good support that will make any MM/1 owner happy. It truly is the soul of a new machine.

The accelerator board can be bought from BlackHawk Enterprises, P.O. Box 10552, Enid, OK 73706-0552, 405-234-2347. Get one-it'll be worth it.

BASIC Programming PART 1 – VARIABLES

By Don Vaillancourt

Don, owner of **CANAWARE,** *has been using a Color Computer for the past eight years or more. Most of his time is spent programming under OS-9 Level II on his CoCo3. His many hobbies include cycling, programming and "BBSing".*

This is the first of a series of articles I will be writing to help computer users to program efficiently. The examples that I will be using revolve around BASIC09, Microware's version of Basic. I find BASIC to be the most powerful multipurpose programming language in the world today. Granted it's not as fast as C or assembly language or quite as intricate as Pascal, but when you look at the number of commands available and their wide variety of uses and flexibility, you begin to wonder why it is only an entry level programming language. Basically, once you master BASIC, your only limiting factor becomes your imagination.

Different people view programming in different ways. Some program to obtain results in their favor, others program because they want to share their ingenuity with others. Then there are the ones that do it for money and still others do it because it becomes an art. I think that these four factors represent programmers very well. Although one attribute may overcome another, over time they seem to change.

VARIABLE DIMENSIONING

In this article I will look at variable dimensioning. Sure, you may think that variable dimensioning is the most boring subject of programming, but often the most boring becomes the most important because it is responsible for the conduct of an operation, which in this case is your program.

One thing that gets overlooked in programming is that variable names in today's languages often support a minimum of 8 significant characters to identify a variable and some may allow up to 32 significant characters, unlike the older programming languages, specifically the ROMed basics, which allowed a maximum of only two.

When defining a variable, use the most definitive word to define its use. Make your variable names as unique as possible and as long as needed, remembering the maximum number of significant characters allowed. Usually, a variable name should be between 8 to 20 characters long, or a full word, but no longer than 32. There are exceptions to this rule, which we'll cover later on.

Also keep in mind that most languages only allow the alphabetical letters (upper and lowercase), numbers, the dollar sign '$' and the underscore '_' to be used as part of a variable name. Some languages, such as COBOL, will allow other characters as well such as the hyphen '-', and other BASIC languages use the percent sign '%' and other deliminator type characters to identify a variable type.

One character that should never be used is the period '.', because this character separates a parent element from its child element in record type variables.

One rule that should never be forgotten is that all variables must start with an alphabetical character. That is a letter from A to Z in upper or lower case.

For example, you may define the variable AGE to store a person's age, or MEN_AGE to store a man's age. Or define the variable NAME to store a person's name or FIRST_NAME and LAST_NAME to store a person's first and last name respectively.

As noted above, the variable name 1COUNT would be illegal. Because the first character is a numeric character, the interpreter or compiler would assume that you want to use a number when in fact you have a variable name, which in turn confuses the interpreter or compiler.

Here are more examples:

```
MONTH12      legal
7_DAYS       illegal, first char is
             a number
name$        legal
My-Name      legal *
#YEARS       illegal, first char is
             not alphabetical
ADDRESS_     legal
HIS_AGE%     legal *
TITLE$STR    illegal, the dollar
             sign '$' must be at the
             end
```

First, check your programming language manual for the variables with the '*' to see if they are supported. If not, then stick with the basics of letters first, numbers, underscore '_' and dollar sign '$'. Just thought I'd note that older basic interpreters did not support the underscore '_'.

VARIABLE TYPES

There are basically five types of variables. They are STRING, BYTE, INTEGER, REAL and BOOLEAN. But be careful because not all programming languages are created equal. Most languages will use the above

five, but some Pascal languages don't use the BYTE definition.

The old ROMed basics did not use any of the five types. They used the dollar sign '$' to identify a string and no dollar sign '$' or maybe the percent sign '%' to identify a real variable.

The COBOL language uses the PICTURE statement and a series of 9's and the letter 'V' to signify a real variable, and more 9's to define the decimal accuracy, the letter 'X' to define a string, or 'A' to define a string with alphabetical characters only.

The STRING definition will cause a variable to be able to store 1 to N characters, from decimal value 0 to 255. By using the single 'quote' or double "quote", you can store printable characters in a string or the CHR$ command to store any character codes between 0 and 255 inclusive.

The BYTE definition is 8 bit large and will hold a value between 0 and 255 inclusive. INTEGER is 2 bytes large or 16 bits and can hold a value between -32768 and 32767. On larger systems and some programming languages it can be 4 bytes large. REAL on the other hand, is five bytes, which is quite large, and again can be up to 10 bytes large, and can hold values from (-1 X 10 -38) to (1 X 10 38).

BOOLEAN types are quite useful; it creates a condition variable. Conditions based upon whether the results of the variable is true or not. A variable defined as BOOLEAN can hold the values TRUE or FALSE only. Although it may seem primitive, it helps a lot to organize one's programming capabilities.

Which ever way your language defines a variable, always remember a variable's maximum storage capacity when defined. Don't try to store a value larger than 255 in a BYTE type variable, or don't hope that a string defined to hold 10 characters will hold 11.

In some languages, there are usually one or two variable types which don't have to be defined, since they are defined automatically when first used. In some cases this may be all right, but I always make sure that I've defined my variables at the top of my program. That way if there is a problem in the program's execution with a variable, I can always check at the top of the listing to see how that variable was defined. Also, it helps others foreign to your programming style and your languages capabilities and rules.

Another way to help your programming is to use the dollar sign '$' at the end of any variables that you define as string. Although not necessary in most high level languages, it can help any programmer who gets lost easily in his programming.

Here are some examples of variable definitions:

```
DIM COUNT:INTEGER  REM NBR OF TURNS TAKEN
DIM NAME$:STRING[32]  REM PERSON'S NAME
DIM MY_AGE:BYTE  REM THIS IS MY AGE
DIM PETS:INTEGER  REM NBR OF PETS
DIM CAPITAL:REAL  REM CAPITAL I INVESTED
DIM FLAG:BOOLEAN  REM TRUE IF DONE, ELSE FALSE
```

If you program with a group of people or you know that your program will be read by others, you might want to explain a variable's use by using the REM statement. That way when the variable is encountered in the program the other person will have an idea of what you're trying to do. It is also a good reminder for yourself as well of what you were doing. A programmer's style evolves over time, especially if he/she didn't know what they were doing at the time of the program's conception.

MULTI-DIMENSION VARIABLES

Multidimensioning variables is easy but sometimes requires large amounts of memory. In physical terms it's hard to explain a multidimensional variable, especially when you get to four and five dimensions and beyond. Let's try. A one dimensional variable can best be described as a straight line cut into several segments from 1 to N. Such as:

```
DIM MONTHS(12):STRING[14]
```

The number between the parentheses represents the number of elements in the variable, which in this case is equal to the number of months in a year, 12. Since each month occurs one after the other, it can be defined as one dimensional, where:

```
MONTHS(1)=JANUARY
MONTHS(2)=FEBRUARY
MONTHS(3)=MARCH
MONTHS(12)=DECEMBER
```

A two dimensional variable could be defined as a Tic-Tac-Toe grid. Using this format:

```
DIM GRID(3,3):STRING[1]
```

A Tic-Tac-Toe grid is made up of 3 columns and 3 rows, where the first digit (called a subscript) of GRID represents columns and the second subscript represents rows. Therefore for each column there are 3 rows.

```
GRID(1,1)=X | GRID(2,1)=O | GRID(3,1)=O
GRID(1,2)=X | GRID(2,2)=X | GRID(3,2)=O
GRID(1,3)=O | GRID(2,3)=O | GRID(3,3)=X
```

A three dimensional variable can be viewed as a Rubik's Cube. The first subscript is the width, the second subscript is the height and the third is the depth. Or in any order easily understood.

```
DIM RUBIK_CUBE(3,3,3):STRING[1]
```

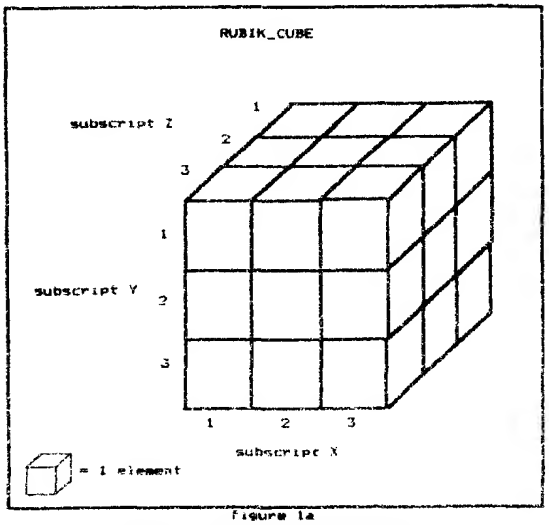


figure 1a
variable cube of RUBIK_CUBE(3,3,3)

Another way to look at the RUBIK_CUBE variable is as a tree. Remember that the first subscript is the parent of the second subscript, and that subscript is the parent of the next subscript and so on, just like a family tree. See figure 1b to get a better idea of what I mean.

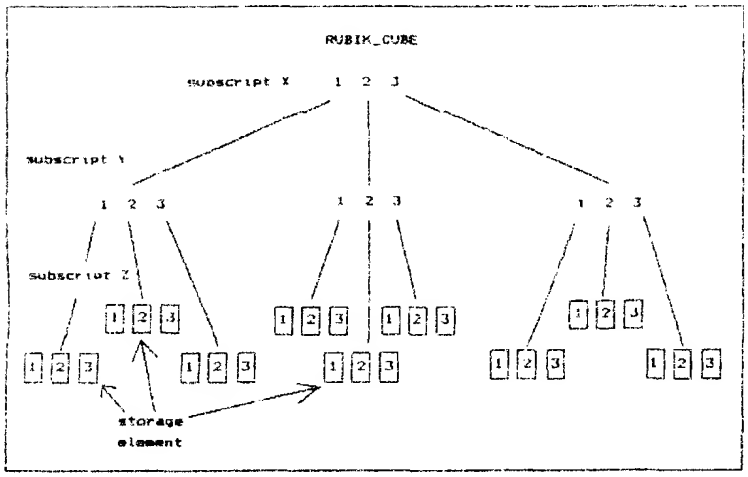


figure 1b
variable tree of RUBIK_CUBE(3,3,3)

Variables can be dimensioned in any size you want with as many dimensions as you need. But some languages are very restrictive to the number of dimensions you can use. And remember, for each dimension you add, you greatly increase the amount of memory by that multiplied dimension value.

One thing you should be aware of is what the first element value of your dimensioned variable is. Some language's first element may start at 0 and some start at 1, and there are some that are adjustable. BASIC09 can be adjusted to be either 0 or 1 by using the BASE statement. When defining a variable in PASCAL, you must specify the first and last subscript of the variable. In the examples above I assumed that the base was 1, but had it have been 0, all subscripts would be 1 less when the variable was addressed.

CREATING RECORD TYPE VARIABLES

Creating record type variables is quite easy. Again, each language is quite different when creating such variables. BASIC uses the TYPE statement, although some of the older ROMed basics didn't have a TYPE statement implementation. PASCAL uses TYPE as well, C uses the statement STRUCT, and COBOL uses incrementing numbers to define a master variable, its elements and those elements' elements.

Record type variables are used when dealing with files for storing or retrieving a set number of records. Also they can be used when transferring lots of mixed information to a subroutine. Here is an example of what a record type variable looks like.

```
TYPE PEOPLE=FIRST_NAME$,LAST_NAME$:STRING[16];
AGE:BYTE;GENDER:BOOLEAN
```

Here the type definition represents four variables, a person's first name, last name, age and then their gender or sex. But since PEOPLE is a type definition, it cannot be used as a variable. PEOPLE will be regarded the same as STRING, BYTE, INTEGER, REAL and BOOLEAN are regarded, as a type definition.

To use PEOPLE it must be defined into a variable and can be used more than once.

```
DIM PERSON:PEOPLE
DIM STUDENT:PEOPLE
```

Now the variables PERSON and STUDENT represent PEOPLE. To access all of PEOPLE's elements you must use the period '.'.

```
PERSON.FIRST_NAME$=DON
PERSON.LAST_NAME$=VAILLANCOURT
PERSON.AGE=23
PERSON.GENDER=TRUE REM IN THIS CASE
                  TRUE MEANS MALE AND
                  FALSE MEANS FEMALE
```

You can do the same with the STUDENT variable and any other variables that were defined with the same type.

One problem with types is that a variable defined in a type cannot be redimensioned as a totally different variable, that is as a stand alone

or part of another TYPE statement. It will cause a variable already dimensioned error such as the example below after the TYPE statement above was defined.

```
DIM FIRST_NAME$:STRING[16]
```

One way that I learned to remedy this problem was in my COBOL class in college. The solution is to use parts of the type definition name in its elements. For the TYPE example above, you could redefine it as:

```
TYPE PEOPLE=PEO_FIRST_NAME$,PEO_LAST_NAME$:STRING[16];
PEO_AGE:BYTE;PEO_GENDER:BOOLEAN
```

One thing that's great about TYPEs is that they can be nested together. It's a great way to organize your data storage and to make your TYPE definitions more readable. Again this technique is used greatly in COBOL. COBOL allows you to group similar variables together, an example follows. By the way, this is one of the reasons why COBOL is still greatly used and why I like it so much.

```
01 PEOPLE-RECORD.
     02 PR-NAMES-PARTS.
          05 PR-NAMES-FIRST-NAME   PIC X(16).
          05 PR-NAMES-LAST-NAME    PIC X(16).
     02 PR-AGE                     PIC 9(02).
     02 PR-GENDER                  PIC A(01).
```

The way you could convert this to BASIC is like this.

```
TYPE NAME_PARTS=NP_FIRST_NAME$,NP_LAST_NAME$:STRING[16]
TYPE PEOPLE_RECORD=PR_NAMES:NAME_PARTS;PR_AGE:BYTE;PR_GENDER:BOOLEAN
DIM PEOPLE:PEOPLE_RECORD
```

To access NP_FIRST_NAME$ or NP_LAST_NAME$, you'd have to first access the first type first and then the second type just like this.

```
LET PEOPLE.PR_NAMES.NP_FIRST_NAME$=DON
```

I didn't want to confuse you before, but here is one of the exceptions to the rule of variable names we discussed earlier. Over time there are non-verbal agreements made between programmers. What I mean by non-verbal is that one programmer picks it up from another programmer's code and so on and so on.

Regarding variable names the exceptions are 'I' and 'J' which represent general purpose incremental and decremental variables for the FOR..NEXT loops as well as REPEAT...UNTIL and other types of conditional loops.

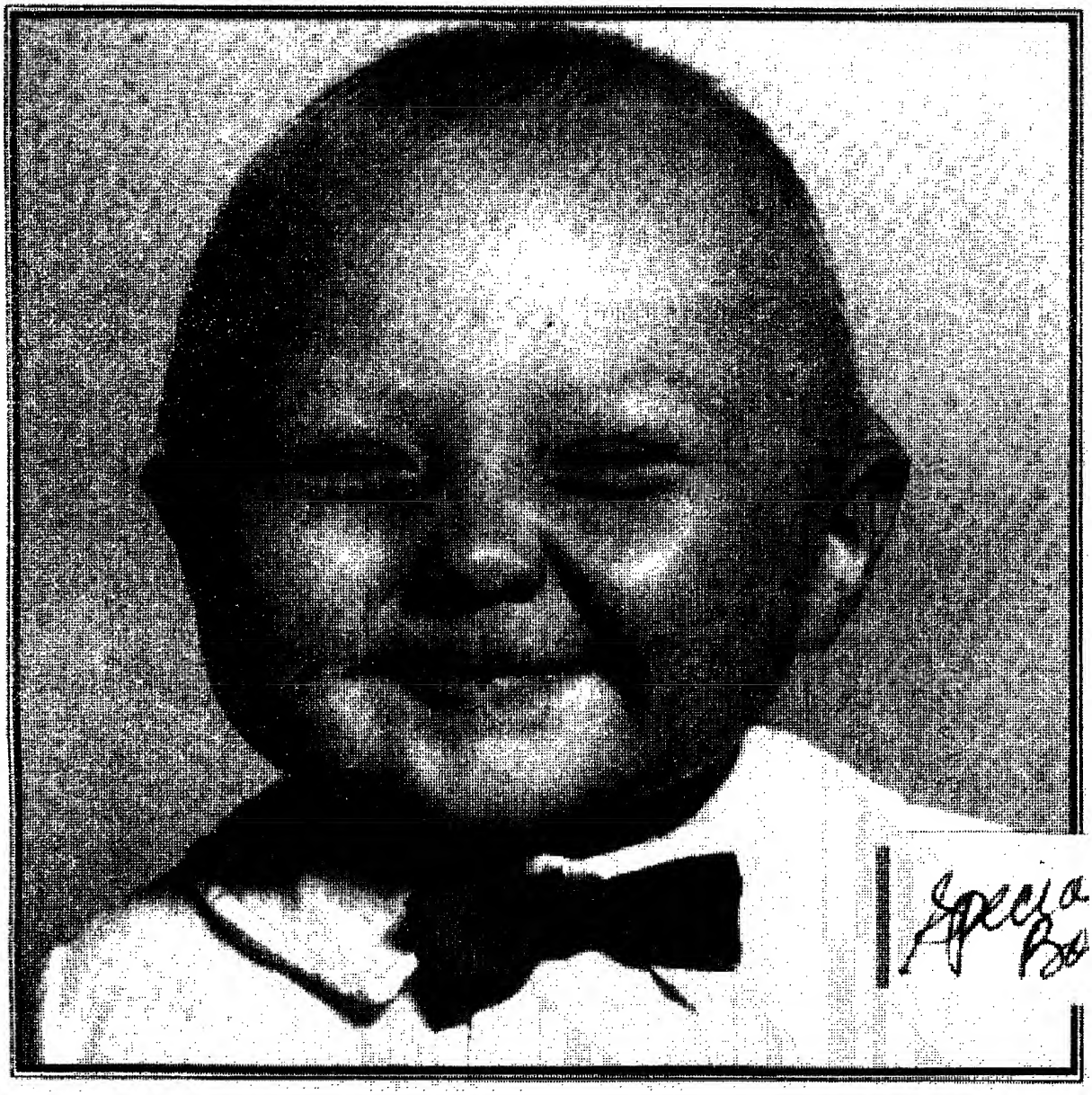
The variables 'X', 'Y' and 'Z' represent their uses as co-ordinates in graphics most of the time or other multidimensional areas.

Sometimes the variable 'C' is used in C (the language) as a means to carry one character from an input function to a larger variable.

There are other exceptions as well, but most of them you will pick up as your programming skills become more proficient.

Anyway, that's it for this month. Hopefully, I'll have another article next month to cover something else, maybe modular programming, or the basic structure of a program.

A happy user!



Dirt Cheap Computer Stuff Company

"Cheap, but not trash"

Announcing Two New Applications for the MM/1

Speedisk Version 1.10 - $99.95

and

CirCad Version 1.0 - $79.95

Speedisk is a disk optimizer/defragmentator for the MM/1 and all OSK machines. Optimize your hard drive in minutes, not days! Supports a full screen display on the MM/1. Commercial versions available soon

CirCad, the most complete electronic circuit designing package for the MM/1, helps you design perfect schematics the first time! Creates a Postscript output file that can be printed on a laser printer or using **Ghostscript,** on your dot-matrix printer.

Please call 314-583-1168 for more details. Also call about our other software and hardware products for OS-9 and OSK, such as *InfoXpress,* Floptical drives, CDROMs, and high speed modems. All orders add $5.00 for shipping. Check or money order only please.

Dirt Cheap Computer Stuff Co.
1368 Old Highway 50 East
Union, Missouri 63084

Address Correction Requested